

Practical Service Oriented Architecture



Technology to automate
decision making and
critical business functions.



Copyright Transparent Logic Inc. 2005 | All Rights Reserved



A Look at Practical Service Oriented Architecture

The term Service Oriented Architecture (SOA) refers to an assortment of services that are able to communicate with each other. These communication capabilities may be as basic as the ability to pass data along to another service, or as complex as coordinating events between other services and the consumer of those services through some underlying connection methodology, usually *Web Services*.

The term “*service*” refers to any self-contained function capable of operating regardless of the state of other services that it may be connected to or communicates with.

Although SOA is a hot IT term these days, the actual concept of providing SOA functionality can be traced back as far as early DCOM and Object Request Brokers (ORB) that followed CORBA specifications.

In the most basic implementation of SOA solutions, a service consumer sends a request message to a service provider using some common communication methodology. Today, XML is the communication *glue* that enables two-way communication in the SOA environment.

Achieving the SOA Environment

A successful SOA environment is one where the service layer is completely separated from all of the other architectural layers so that the service consumers communicate with the services layer which then communicate with the business objects.

This arrangement enables the services consumer to indirectly interface with the business objects without the consumer having to possess any knowledge of how the business objects are constructed or what the underlying business language or database is. This means that any consumer capable of working with XML is capable of working with the services.

Advantages of Implementing SOA

1. Better ROI

The implementation of a SOA environment enables developers to put all of the business logic into a separate layer rather than incorporating it into the

code. This enables the logic layer to exist independently of the applications which consume the services and to be immediately available to new versions of the consuming applications without requiring redevelopment of the business logic layer.

2. Better Code Mobility

The ability to lookup and dynamically bind to a service means that services can be located on different servers than the ones that the consumers are hosted on. This provides the organization with the ability to build enterprise-wide solutions hosted in diverse locations both within and outside of the organization.

3. Better Usage of IT Talent

Because the SOA environment uses multiple layers, the organization can assign developers with specific skill sets to work within specific layers. This provides a means to deploy the most qualified people to work in specific roles without regard to the technical skills required to support development within other layers.

4. Enhanced Security

The existence of the SOA service layers result in the creation of additional network interfaces capable of being accessed by multiple applications. In a client-server environment, security is addressed solely at the application's entry point, and vulnerabilities often exist in areas such as databases due to the difficulty in maintaining multiple security lists. By their very nature, services have built-in security mechanisms that allow for multi-level security at the service and the client levels.

5. Ease of Testing and Reduced Defects

Because services have published interfaces, unit tests can be easily written to validate performance before the services are exposed to the consumers. This provides a way to identify and correct defects before the actual application undergoes the QA testing process.

6. Support for Multiple Client Types

The SOA allows diverse client types to access the services using their native communication capabilities including HTML, XML, RMI, etc.

7. Ease of Maintenance

Because all of the logic is encapsulated within the service layer, it is faster and easier for developers to isolate and correct defects as well as add functionality without rebuilding the module from the ground up.

8. Realization of Code Reuse

Incompatibilities between different platforms, clients and languages has kept the dream of substantial code reuse beyond the reach of most organizations.

The SOA environment makes code reuse a reality because consumers are able to discover a service and bind to it without being concerned about languages, compiler versions, or other issues that normally impact code reuse in non-SOA environments.

9. Enhanced Scalability

The SOA environment is location-transparent. As a result, service requests can be diverted by load balancers without affecting communications between the client and the service.

10. Higher Availability

Location transparency also provides the ability for multiple servers to run multiple instances of the same service. This enables services requests to be redirected to other instances or machines in the event of a network segment or machine failure.

Conclusion

Although the SOA approach takes time and planning to implement, the payoff comes in the form of time and cost savings, increased reliability and security, and all of the other benefits previously discussed.